Monitoramento do Consumo de Energia e Acionamento Remoto de Equipamentos por Meio de Redes de Sensores Sem Fio

Ivan de Oliveira Nunes

[1]Universidade Federal do Espírito Santo
2014
Vitória – ES – Brasil

ivanolive@dcc.ufmg.br

Resumo. *A crescente demanda energética no Brasil e no mundo é, atualmente, tema de pesquisa e discussão nas mais diversas áreas, entre elas a engenharia. Motivado pela necessidade de criação de sistemas que permitam analisar e controlar o consumo de energia de equipamentos e viabilizado pelo crescente avanço nas tecnologias de sistemas embarcados, este trabalho propõe um sistema baseado em Wireless Sensor Networks (WSN) que permite realizar, remotamente, o monitoramento e o acionamento de equipamentos eletrônicos diversos, seja em redes locais (LAN) ou por meio da Internet. Como prova de conceito, foi implantada uma WSN com uma central de controle e dois nodos sensores/atuadores para controlar e monitorar equipamentos típicos de ambientes residenciais por meio de uma interface Web e de um aplicativo para dispositivos móveis.*

UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
CENTRO TECNOLÓGICO
DEPARTAMENTO DE INFORMÁTICA
PROJETO DE GRADUAÇÃO

IVAN DE OLIVEIRA NUNES

Monitoramento do Consumo de Energia e Acionamento Remoto de Equipamentos por Meio de Redes de Sensores Sem Fio

VITÓRIA

2014

IVAN DE OLIVEIRA NUNES

Monitoramento de Consumo de Energia e Acionamento Remoto de Equipamentos por Meio de Redes de Sensores Sem Fio

Monografia apresentada como Projeto de Graduação em Engenharia de Computação da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do grau de Engenheiro de Computação.

Orientador: Prof. Dr. Magnos Martinello

VITÓRIA

2014

IVAN DE OLIVEIRA NUNES

Monitoramento do Consumo de Energia e Acionamento Remoto de Equipamentos por Meio de Redes de Sensores Sem Fio

Monografia apresentada como Projeto de Graduação em Engenharia de Computação da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do grau de Engenheiro de Computação.

BANCA EXAMINADORA

Prof. Dr. Magnos Martinello - Orientador
Universidade Federal do Espírito Santo

Prof. Dr. Maxwell Eduardo Monteiro
Instituto Federal do Espírito Santo

M.Sc. Saymon Castro de Souza
Universidade Federal do Espírito Santo

Vitória, _____ de _____ de _____.

AGRADECIMENTOS

Agradeço primeiramente a Deus por me dar força para lutar pelos meus sonhos, sempre recompensar meus esforços e por ter colocado no meu caminho pessoas incríveis que cito nestes agradecimentos.

Aos meus pais, João e Angélica, por me darem amor, carinho e apoio incondicional. Por me ensinarem a ser homem. Por me mostrar o valor da verdade, da humildade, da amizade e do amor. Por me explicar que “A caridade é a mãe de todas as qualidades e a soberba a mãe de todos os defeitos”.

Ao meu irmão, Igor, pela amizade, confiança e companheirismo. Por todas as brincadeiras e brigas de nossa infância inesquecível. Por sempre perder para mim em todo e qualquer jogo de vídeo-game =P.

A todos os familiares que mesmo distantes se fazem presentes em minha vida. Em especial a meus avós, tios e primos.

Aos meus amigos de infância: Vitor, Karison, João Afonso, e Botox. Por me apoiarem e entenderem minha ausência completa de vida social, exigida pelo curso de Engenharia de Computação.

Aos amigos de intercâmbio pela companhia durante o ano mais incrível da minha vida. Por, em um ano de convivência, se tornarem amigos para a vida toda.

À minha namorada, Renata, por ser minha força nos momentos de dificuldade, por sempre me ouvir e me tranquilizar. Por ser companheira, amiga e carinhosa. Por me ligar e me mandar escrever o PG quando a fanfarra tomava conta do meu espírito =D.

Aos amigos de faculdade, em especial à turma 09: sem dúvida a melhor turma de Engenharia de Computação de todos os tempos. *“Ô zero 9, zero 9! A tua estrela vai brilhar! Tua história é tua glória! ...”*.

Aos professores, por me passarem o conhecimento e me tornarem um profissional competente. Especialmente à Roberta e ao Magnos, pela ajuda prestada, não só como orientadores/tutora, mas em diversos momentos fundamentais da minha

trajetória acadêmica. Tenho ambos como modelos de profissionais nos quais eu quero me espelhar no exercício da minha profissão.

À ERUS e ao PET EngComp por contribuírem enormemente para minha formação como profissional e cidadão. E pelos amigos que fiz nessas organizações.

Enfim, a todos que de alguma forma contribuíram para que eu pudesse chegar até aqui.

A todos, meu muito obrigado!

“Looking forward for the next life adventure.”

RESUMO

A crescente demanda energética no Brasil e no mundo é, atualmente, tema de pesquisa e discussão nas mais diversas áreas, entre elas a engenharia. Motivado pela necessidade de criação de sistemas que permitam analisar e controlar o consumo de energia de equipamentos e viabilizado pelo crescente avanço nas tecnologias de sistemas embarcados, este trabalho propõe um sistema baseado em *Wireless Sensor Networks* (WSN) que permite realizar, remotamente, o monitoramento e o acionamento de equipamentos eletrônicos diversos, seja em redes locais (LAN) ou por meio da Internet. Como prova de conceito, foi implantada uma WSN com uma central de controle e dois nodos sensores/atuadores para controlar e monitorar equipamentos típicos de ambientes residenciais por meio de uma interface Web e de um aplicativo para dispositivos móveis.

Palavras-chave: WSN, Sistemas Embarcados, Automação Residencial, Sensoriamento Remoto, Zigbee.

LISTA DE FIGURAS

Figura 2-1: Exemplo básico de uma WSN. 15
Figura 2-2: Arquitetura da WSN implementada. 17
Figura 2-3: Componentes dos nodos sensores. 18
Figura 2-4: Componentes da central de controle. 19
Figura 2-5: Interface do *software* X-CTU. 21
Figura 2-6: Exemplos de topologias implementadas com módulos Xbee. (FALUDI, 2011) 23
Figura 2-7: Nodos agrupados de acordo com canal, PAN ID e NODE ID. (FALUDI, 2011) 24
Figura 3-1: Componentes do *hardware* do *Raspberry Pi*. 28
Figura 3-2: Central da WSN implementada com o *Raspberry Pi*. 29
Figura 3-3: Acesso a site hospedado no *Raspberry Pi*. 30
Figura 3-4: Configuração da tabela de encaminhamento do roteador. 31
Figura 3-5: Obtenção do ID da porta serial a ser utilizada. 33
Figura 3-6: Abertura da conexão serial. 34
Figura 3-7: Recepção dos dados com o método *serialEvent*. 35
Figura 3-8: Arduino UNO e suas interfaces. 37
Figura 3-9: Interface do Arduino com o Xbee. 38
Figura 3-10: Sensor de corrente TA12-100. 39
Figura 3-11: Circuito de acionamento ligado ao Arduino. 40
Figura 3-12: Máquina de estados implementada no Software do Arduino. 42
Figura 3-13: Configuração da interrupção por tempo no Arduino. 43
Figura 3-14: Onda senoidal amostrada. 44
Figura 3-15: Vetor de medições obtido por requisição HTTP. 45
Figura 4-1: Site que mostra o consumo de energia dos aparelhos. 48
Figura 4-2: Aplicativo para monitorar e controlar os equipamentos. 49
Figura 4-3: Acionamento da lâmpada no segundo 78 e do ventilador no segundo 83. 50
Figura 4-4: Acionamento da lâmpada no segundo 35 e desligamento do ventilador no segundo 45. . 51
Figura 4-5: Ventilador (consumo 80W) mantido ligado. Lâmpada (consumo 30W) sendo desligada no segundo 87. 52
Figura 4-6: *Setup* do *hardware* de sensoriamento e acionamento em uma lâmpada. 53

LISTA DE SIGLAS/ACRÔNIMOS

API – Application Programming Interface

DDNS – *Dynamic Domain Name Server*

DNS – *Domain Name Server*

GPIO - *General Purpose Input Output*

HDMI - *High-Definition Multimedia Interface*

HTTP - *Hypertext Transfer Protocol*

LAMP – Linux, Apache, MySQL, PHP

LAN – *Local Area Network*

MAC – *Media Access Control*

NAT – *Network Address Translation*

PAN – *Personal Area Network*

PHP - PHP*: Hypertext Preprocessor*

PWM – *Pulse-width Modulation*

RSSF – Rede de Sensores Sem Fio

USART - *Universal Synchronous Asynchronous Receiver Transmitter*

WSN – *Wireless Sensor Networks*

XML – *eXtensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO 10

1.1 OBJETIVOS 11

1.1.1 Objetivos Gerais 11

1.1.2 Objetivos Específicos 11

1.2 METODOLOGIA 12

1.3 ESTRUTURA DO PROJETO DE GRADUAÇÃO 12

2 *WIRELESS SENSOR NETWORKS* 14

2.1 INTRODUÇÃO ÀS *WIRELESS SENSOR NETWORKS* 14

2.2 CARACTERIZAÇÃO DA WSN DESTE PROJETO 15

2.3 ARQUITETURA DA WSN IMPLEMENTADA 17

2.4 DEFINIÇÃO DA TOPOLOGIA DA REDE 19

3 SISTEMAS EMBARCADOS 26

3.1 DESCRIÇÃO E SELEÇÃO DOS SISTEMAS EMBARCADOS 26

3.2 *RASPBERRY PI* E A CENTRAL DA WSN 27

3.3 ARDUINO E OS NODOS SENSORES 36

4 IMPLANTAÇÃO E TESTES 46

4.1 DESCRIÇÃO 46

4.2 REDE IMPLANTADA 46

4.3 INTERFACE COM O USUÁRIO 47

4.4 TESTES 49

5 CONCLUSÃO 54

5.1 TRABALHOS FUTUROS 54

6 REFERÊNCIAS BIBLIOGRÁFICAS 57

1 INTRODUÇÃO

Atualmente há um aumento na demanda energética em todo o mundo. Esse aumento se dá, dentre outros motivos, pelo crescente número de equipamentos eletrônicos de alto consumo em ambientes residenciais e pela necessidade das grandes indústrias manterem o funcionamento de equipamentos de grande porte para atender a um mercado cada vez mais exigente. Devido a esse crescimento, o fornecimento de energia torna-se, a cada dia, um desafio ainda maior.

O projeto de pesquisa "*Software Defined Buildings (SDB)*", da *University of California in Berkley*, (DAWSON-HAGGERTY et al., 2012) define como uma das diretrizes maiores dos prédios do futuro a criação de mecanismos que possibilitem o controle do consumo energético. Neste contexto, a proposta de soluções inovadoras que permitam avaliar o consumo e viabilizar a economia de energia passam a ter um papel fundamental.

Este projeto vem sendo desenvolvido ao longo do último ano e tem por objetivo definir e implantar uma arquitetura de *Wireless Sensor Networks* (WSN) que permita monitorar o consumo de energia de um conjunto de equipamentos de interesse. Adicionalmente, o sistema deve permitir ligar e desligar os equipamentos desejados. Ao usuário do sistema deve ser provida a funcionalidade de acessá-lo por meio de qualquer computador ou dispositivo portátil (*smartphone* ou *tablet*) conectado à internet ou que se encontre dentro da mesma rede local, LAN, onde está instalada a central de controle do sistema.

O foco deste trabalho está na economia de energia, porém, o conjunto de funcionalidades implementadas pode ser facilmente estendido, mediante algumas adaptações de hardware, para que, usando a mesma arquitetura (a ser descrita nos próximos capítulos) sejam contemplados casos de uso típicos de sistemas de automação residencial, como controle de dispositivos multimídia, climatização, controle de iluminação e segurança residencial. Tudo em um mesmo sistema, sem grandes custos adicionais.

Uma questão central no projeto de sistemas como esse é sua viabilidade econômica. Há alguns anos atrás, a implantação de um sistema como o proposto neste trabalho seria inviável devido ao custo do hardware necessário. Graças à grande evolução na tecnologia de sistemas embarcados, hoje é possível construir um sistema que, em um curto período de tempo, proporcione uma economia que supere o valor investido em sua implantação, não só quando se trata de grandes indústrias, mas também no caso das residências. Além disso, são inegáveis os benefícios ambientais e sociais advindos do racionamento e de um uso consciente da energia elétrica.

1.1 OBJETIVOS

Os objetivos deste projeto de graduação podem ser divididos em objetivos gerais e específicos como definidos a seguir.

1.1.1 Objetivos Gerais

Definir, implementar e implantar uma WSN capaz de monitorar o consumo de energia, além de ligar e desligar, aparelhos eletrônicos em ambientes domésticos, comerciais ou industriais.

Além disso, tem-se como objetivo geral a obtenção de uma experiência com as tecnologias envolvidas, dentre elas: WSN, programação de micro controladores, configuração de provedores de serviços WEB e desenvolvimento de interfaces amigáveis com o usuário.

1.1.2 Objetivos Específicos

- Implementação do dispositivo central da WSN utilizando sistema embarcado *Raspberry Pi* juntamente com os dispositivos de comunicação *Xbee*.

- Implementação dos módulos sensores e atuadores por meio do sistema embarcado de prototipação *Arduino* em conjunto com os dispositivos de comunicação *Xbee*.
- Criação de uma interface Web para monitoramento e controle dos equipamentos eletrônicos por parte do usuário do sistema implantado.
- Validação da arquitetura proposta, com a implantação de uma WSN de dois módulos sensores/atuadores com capacidade de ligar/desligar equipamentos e monitorar o consumo de energia de cada um dos módulos.

1.2 METODOLOGIA

Para alcançar os objetivos propostos foi utilizada a seguinte metodologia:

1. Estudo sobre os sistemas embarcados *Raspberry Pi* e *Arduino* e sobre o hardware de comunicação *Xbee*;
2. Implementação do protocolo de comunicação da WSN;
3. Implementação do hardware e do software dos módulos de sensoriamento e acionamento usando o hardware de prototipação *Arduino*;
4. Implementação do software de aquisição e disponibilização dos dados da WSN no dispositivo central de controle e monitoramento, *Raspberry Pi*;
5. Configuração de um servidor LAMP no *Raspberry Pi*, possibilitando o acesso via WEB aos recursos do sistema;
6. Implantação do protótipo do sistema implementado;
7. Validação do protótipo por meio de testes de casos de uso.

1.3 ESTRUTURA DO PROJETO DE GRADUAÇÃO

Este Projeto de Graduação tem em seus primeiros capítulos uma abordagem teórica sobre as WSN, os sistemas embarcados, o hardware e os protocolos utilizados no desenvolvimento. Com isso, servem de embasamento para os capítulos seguintes.

O Capítulo 2 explicita os conceitos básicos das WSN, expondo suas principais características, arquiteturas e funcionamento. São apontadas e justificadas as

decisões de projeto e descrita em detalhes a implementação da WSN utilizada com o *Hardware* de comunicação *Xbee*.

O Capítulo 3 discorre de forma simplificada sobre o funcionamento dos sistemas embarcados *Arduino* e *Raspberry Pi*. São ainda descritas as implementações de *Hardware* e *Software* realizadas em cada um destes módulos, responsáveis pelo funcionamento do projeto como um todo.

O Capítulo 4 explica a implantação da central e dos módulos sensores/atuadores em um cenário de testes. É introduzida a interface criada para interação com o usuário e são mostrados testes do funcionamento do sistema.

O Capítulo 5 traz as conclusões deste Projeto de Graduação. São apresentados os resultados obtidos e as propostas de trabalhos futuros.

2 *WIRELESS SENSOR NETWORKS*

Este capítulo dá uma visão geral sobre as WSNs, seus principais elementos e características. É definida a WSN implementada neste Projeto de Graduação e descrita sua arquitetura e funcionamento.

2.1 INTRODUÇÃO ÀS *WIRELESS SENSOR NETWORKS*

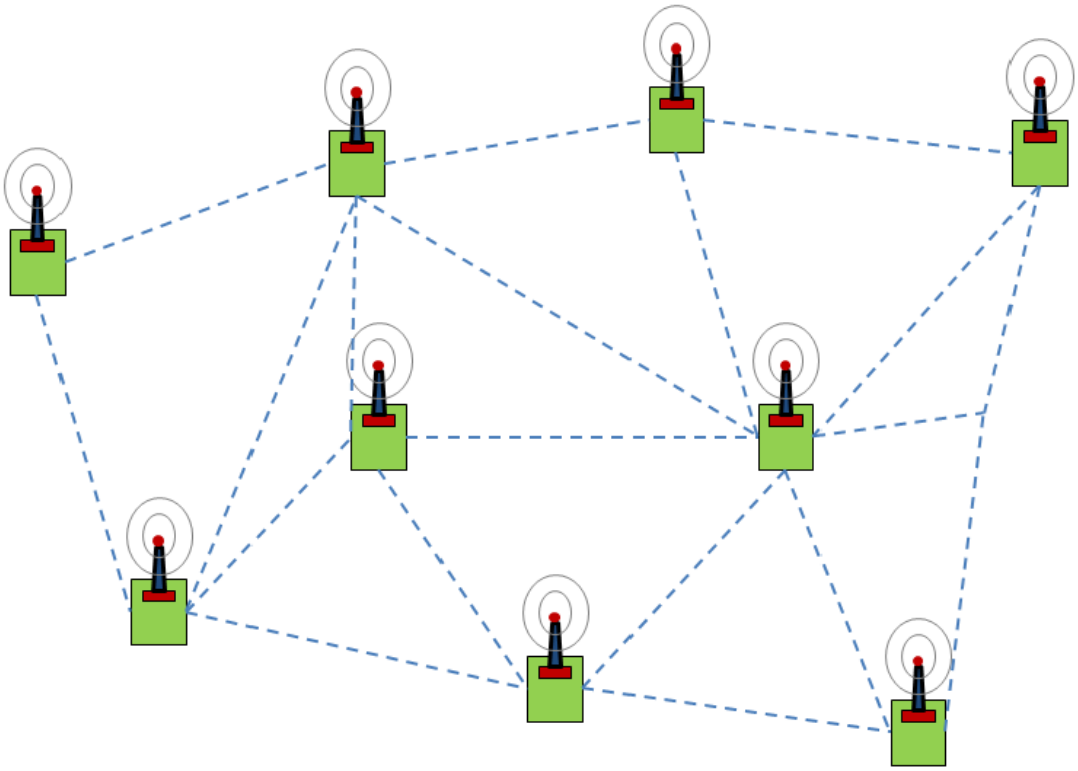
O grande avanço que vem ocorrendo nas últimas décadas na área de microprocessadores, sensores/atuadores e nos protocolos de comunicação sem fio tem propiciado o desenvolvimento de sistemas inteligentes de sensoriamento de grandezas físicas, químicas, biológicas, entre outras. Esse tipo de sistema de sensoriamento tem vastas aplicações nas mais diversas áreas do conhecimento. Sistemas desse tipo são conhecidos como *Wireless Sensor Networks, Wireless Sensor and Actuators Networks* (WSAN) ou Redes de Sensores Sem Fio (RSSF).

As *Wireless Sensor Networks* diferem de redes de Redes de Computadores em diversos aspectos. Essas redes se distinguem por possuírem um grande número de nodos (ou sensores) distribuídos, restrições de consumo de energia, e necessidade de mecanismos para autoconfiguração e adaptação devido a problemas com falhas e perda de nodos. Por isso os algoritmos e protocolos definidos para as redes de computadores tradicionais precisam ser revistos quando se trata de uma rede deste tipo (LOUREIRO, 2006).

O componente básico de uma WSN é o nodo sensor, um elemento computacional com capacidade de processamento, memória, interface de comunicação sem fio, um ou mais sensores e eventualmente atuadores, quando se deseja controlar algo no ambiente da aplicação (como no caso deste trabalho). A ideia é que os nodos sensores tenham tamanho e custo reduzido e capacidade computacional suficiente para monitorar os dados desejados e transmiti-los com sua interface de comunicação. Dessa forma, torna-se possível utilizá-los em larga escala, ou seja, espalhá-los no ambiente a ser monitorado e/ou controlado (SILVA et al., 2004).

A WSN é formada por um conjunto de nodos sensores que se intercomunicam de modo a promover o espalhamento dos dados desejados. A figura 2-1 exemplifica uma rede de sensores básica.

Figura 2-1: Exemplo básico de uma WSN.



2.2 CARACTERIZAÇÃO DA WSN DESTE PROJETO

Como o conceito de WSN pode ser utilizado em diversos domínios de aplicação, a arquitetura deve ser projetada para atuar da forma mais eficiente de acordo com o problema que se deseja solucionar. De acordo com RUIZ (2003), as WSN podem ser classificadas segundo a configuração, o tipo de sensoriamento e o tipo de comunicação. Para definir a classificação da WSN a ser projetada, deve-se primeiramente definir o objetivo do sistema e identificar as restrições da WSN quando aplicada a esse tipo de sistema.

No contexto deste Projeto de Graduação, a WSN tem por objetivo monitorar o consumo de energia de equipamentos eletrônicos e dar ao usuário deste sistema a capacidade de ligar ou desligar estes equipamentos remotamente. Isso possibilitaria um uso consciente e econômico da energia elétrica. Partindo desta premissa, primeiramente são apontadas as seguintes restrições:

1 – Restrição de energia nos nodos: Para o monitoramento do consumo de energia não há preocupação com bateria, já que os nodos estarão conectados à instalação elétrica do ambiente. Porém, ainda é necessário que eles consumam pouca energia, caso contrário a implantação do sistema trará mais gasto do que economia de energia.

2 – Restrição de centralidade dos dados: Para possibilitar o acesso remoto à WSN é preciso que pelo menos um dos nodos tenha capacidade de conexão com a rede IP. O nodo que é acessado pela rede IP deve também ter nele todos os dados de consumo de energia, além da capacidade de enviar (direta ou indiretamente) comandos para todos os demais nodos. Por isso é natural eleger um dos nodos para ser o nodo central da rede, ou seja, o nodo ao qual os demais se reportam.

Tendo em vista as características objetivadas e as restrições enumeradas acima, classifica-se a WSN deste projeto:

Quanto à configuração:

- Centralizada/Hierárquica: levando-se em conta a restrição 2, supracitada, o nodo que realiza a interface com a rede IP assume também o papel de coordenador da WSN.

- Homogênea: os nodos possuem a mesma capacidade de Hardware.

- Estacionária: os nodos não sofrem alteração de posição ao longo do funcionamento do sistema.

Quanto ao tipo de sensoriamento:

- Periódico: para a medição do consumo de energia os nodos fazem a aquisição, o tratamento e o envio dos dados em intervalos de tempo pré-definidos.

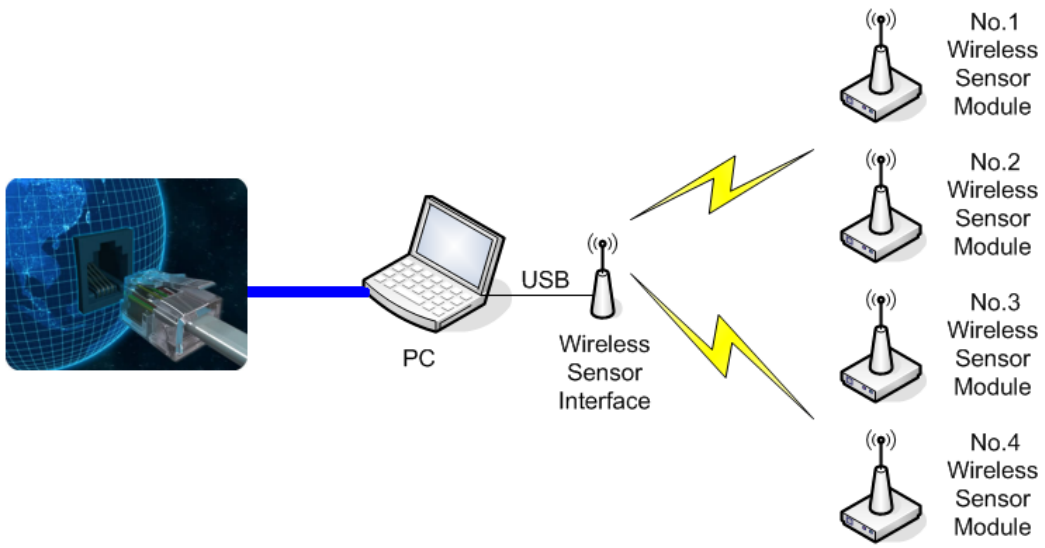
Quanto ao tipo de comunicação:

- Protocolo Zigbee: devido à restrição número 1, apresentada acima, foi escolhido o protocolo de comunicação Wireless Zigbee. Dentre as características do Zigbee se destaca o baixo consumo de energia para sua utilização se comparado a outros protocolos como WiFi, Bluetooth e UWB (POTHUGANTI; CHITNENI, 2014), além de mecanismos intrínsecos de configurabilidade automática de rede e diversas vantagens implementacionais que serão evidenciadas na seção 2.4, se mostrando a melhor alternativa dentre os demais protocolos de comunicação sem fio. Nos trabalhos futuros é discutida ainda a possibilidade de implementação do protocolo IPV6 sobre a camada de enlace Zigbee, de acordo com o padrão 6LOWPAN, dentro do contexto de *Internet of Things*.

2.3 ARQUITETURA DA WSN IMPLEMENTADA

Tendo em vista a caracterização realizada propõe-se a arquitetura mostrada na figura 2-2.

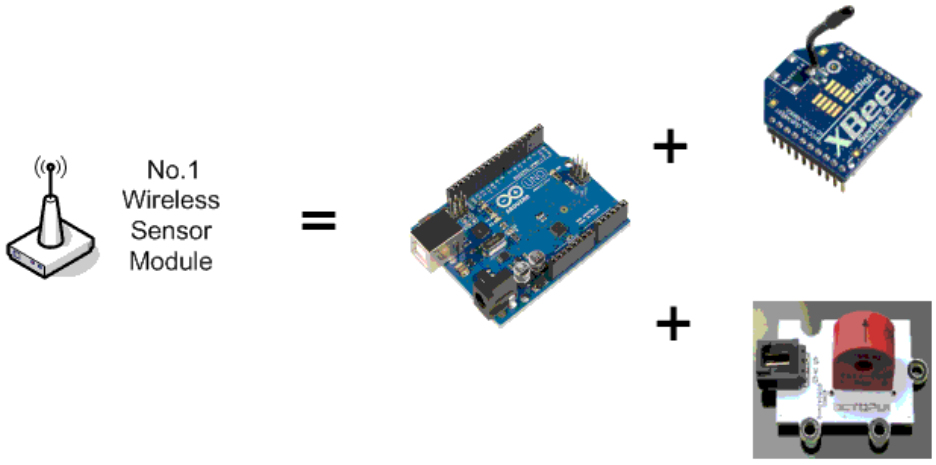
Figura 2-2: Arquitetura da WSN implementada.



De acordo com a figura 2-2, cada *Wireless Sensor Module*, representa um módulo de medição de consumo de energia e ligamento/desligamento de equipamento. Na

perspectiva da WSN, os *Wireless Sensor Modules* são os nodos. Na perspectiva do *Hardware*, cada um deles é um conjunto formado por um sensor de corrente, um sistema embarcado *Arduino*, um dispositivo de comunicação wireless que utiliza o protocolo Zigbee, o *Xbee*, e um circuito de acionamento elétrico baseado em relés. Esse conjunto é mostrado na figura 2-3. Por fim, sob a ótica do *Software*, cada um implementa uma máquina de estados para gerenciar o compartilhamento do processador entre a realização de todas as tarefas necessárias para o seu devido funcionamento. Mais detalhes da implementação dos nodos serão dados no capítulo 3.

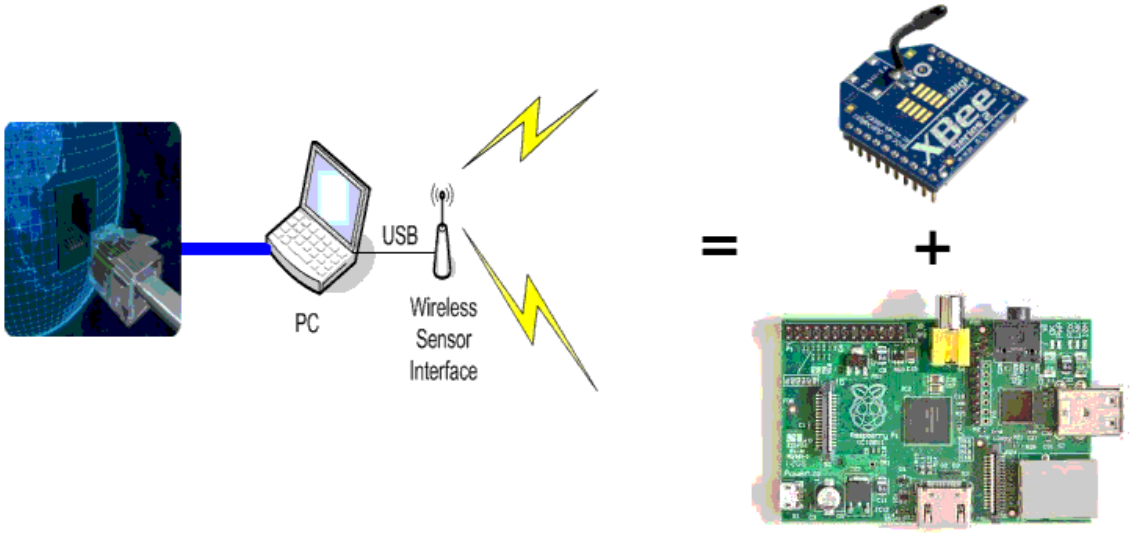
Figura 2-3: Componentes dos nodos sensores.



Ainda na figura 2-2, o *Wireless Sensor Interface* é o módulo da WSN que realiza a interface, por meio de comunicação USB, com a central de controle do sistema. A central de controle do sistema por sua vez se conecta à rede IP, o que é fundamental para que seja possibilitada a utilização do sistema, seja via LAN ou via Internet, por parte de usuário. Um ponto estratégico na implementação deste projeto é a substituição de um *PC Desktop* por um *Raspberry Pi*. Essa substituição possibilita uma economia enorme, além de dedicar um dispositivo exclusivo para a central de controle, o que seria dificultado caso fosse usado um *Desktop*. A figura 2-

4 mostra os componentes da central de controle. Sua implementação será detalhada no capítulo 3.

Figura 2-4: Componentes da central de controle.



2.4 DEFINIÇÃO DA TOPOLOGIA DA REDE

Como descrito anteriormente, nesta implementação os módulos Xbee são os responsáveis pela comunicação Wireless dos nodos. Porém, suas funcionalidades vão muito além de simples emissores/receptores de radiofrequência. Com estes módulos é possível definir a topologia da rede de sensores, atribuindo funcionalidades especificas aos nodos da rede e definindo endereços de receptores/destinatários das mensagens enviadas por cada nodo. Eles são capazes ainda de realizar o roteamento dos dados de forma transparente para quem os utiliza para a transmissão de dados.(DIGI INTERNATIONAL, 2009)

Os módulos Xbee são produzidos pela empresa americana *Digi International* e são amplamente utilizados por desenvolvedores no mundo todo, principalmente pela facilidade de utilização. Devido a essa facilidade o Xbee se consolidou como uma das melhores alternativas para implementação de protótipos. Muito embora se trate de um sistema robusto e funcional, quando há intenção de transformar o protótipo

em um produto, o Xbee deixa de ser tão adequado devido a seu custo relativamente elevado. O preço de um módulo Xbee, varia de $21,00 a $32,00 dólares americanos, variando de acordo com o alcance da transmissão e a quantidade de funcionalidades do módulo. Este valor pode onerar muito o custo final de um nodo sensor. Por isso, muitas empresas optam por utilizar outras soluções de comunicação *Zigbee* que, apesar de não serem tão facilmente utilizáveis, apresentam custos mais acessíveis, permitindo a fabricação de um produto final mais barato e competitivo. Exemplos de empresas que oferecem módulos de comunicação *Zigbee* alternativos ao Xbee são a Atmel e a Sparkfun.

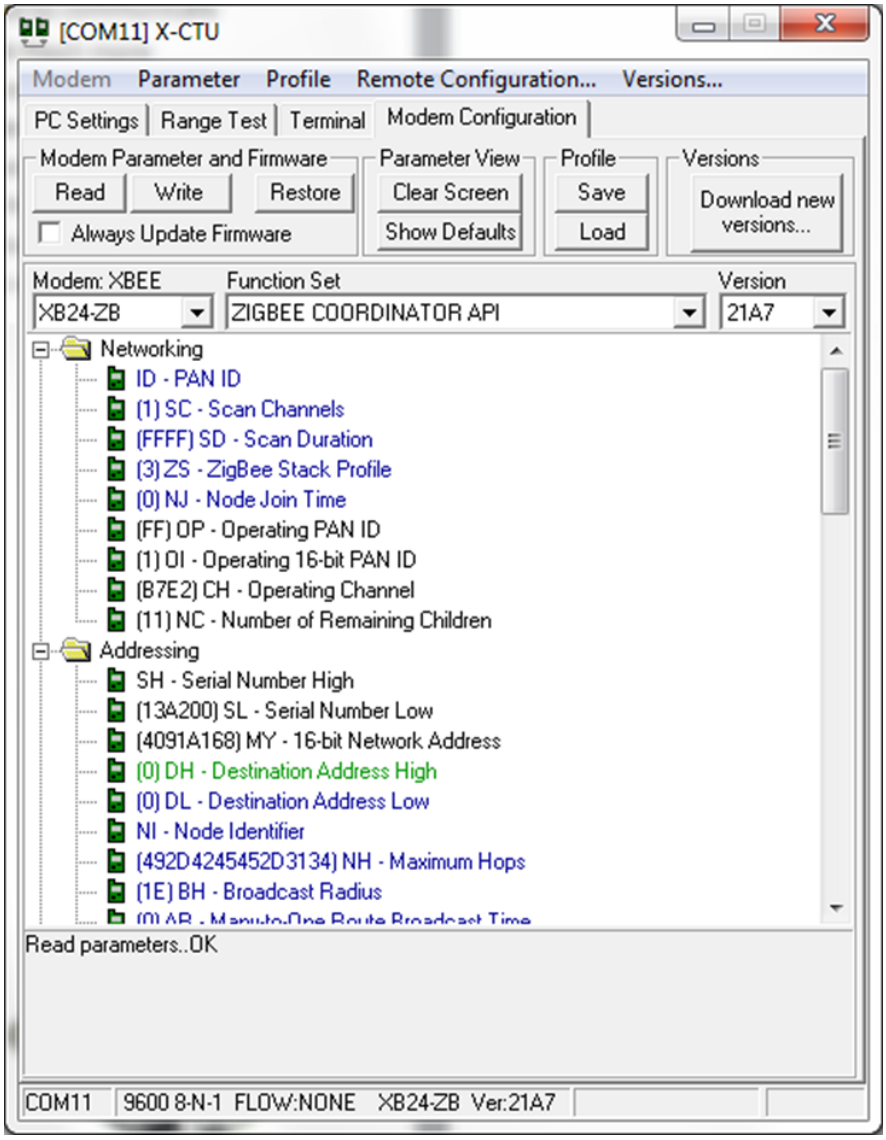
Os módulos Xbee possuem dois modos de utilização, o modo transparente e o modo *API*. O modo *API* possibilita que o desenvolvedor tenha um maior controle sobre o funcionamento do Xbee em si, interferindo em questões como roteamento, encaminhamento de mensagens, garantia ou não da entrega de pacotes, segurança e criptografia, entre outros. Neste projeto, por simplicidade, foi utilizado o modo transparente. Para maiores informações sobre a utilização dos módulos Xbee para construção de WSNs recomenda-se a leitura do livro "*Building Wireless Sensor Networks*" (FALUDI, 2011).

Para definir a topologia da WSN é necessário configurar cada um dos módulos Xbee que serão conectados a cada um dos micro controladores. A Digi Internacional provê, juntamente com os módulos Xbee, uma interface de programação dos módulos denominada X-CTU (figura 2-5). O X-CTU permite configurar aspectos do funcionamento do Xbee relativos à rede como canal de operação, momento de entrada do módulo na rede, entre outros. Além disso, permite também configurar o endereçamento do módulo. Esse último tipo de configuração é mais relevante para a construção da WSN. A programação é realizada via USB usando a placa de prototipação *Digi Evaluation Board*. Em suma, para criar a WSN, os módulos Xbee devem ser configurados quanto ao canal de operação, quanto à função na WSN, quanto à WSN a que pertencem, quanto ao ID do nodo e quanto ao ID do destinatário.

Classificação quanto ao canal de operação:

A classificação quanto ao canal de operação define a faixa de frequências em que serão transmitidos os dados. Embora duas WSNs, ou outros tipos de comunicação wireless possam eventualmente coexistir no mesmo canal de operação (mesmo espectro de frequências) é recomendável que a escolha do canal seja feita de modo a balancear as comunicações simultâneas entre os canais disponíveis, evitando interferências, que podem provocar erros na transmissão dos quadros.

Figura 2-5: Interface do *software* X-CTU.



Classificação quanto à função na WSN:

Um módulo Xbee pode ter, na WSN, uma das três funções: *coordinator, router* ou *end-device*. Embora os três tipos possam enviar e receber mensagens, eles diferem em seus papéis na organização da rede.

Cada WSN deve ter apenas um *coordinator*, que é o responsável por organizar a estrutura da mesma. Ele é o dispositivo central ao qual os demais se conectam e se reportam ao longo do funcionamento da WSN. O *coordinator* é capaz de se conectar diretamente a um ou mais dispositivos.

O *router* é o dispositivo secundário da rede que, além de enviar e receber mensagens, é capaz de encaminhar mensagens de um remetente a um destinatário. Assim como o *coordinator*, ele é capaz de se conectar a um ou mais dispositivos.

Finalmente, o *end-device* é o dispositivo mais simples da WSN. Só tendo capacidade de se conectar diretamente a um dispositivo, ele precisa necessariamente estar conectado a um *router* ou a um *coordinator* para se juntar à WSN.

Todos os modelos de Xbee implementam as funcionalidades de *coordinator* e *end-device,* bastando definir, por meio do software X-CTU, o papel do dispositivo na WSN. Devido à sua maior complexidade, a função de *router* é disponibilizada apenas em alguns modelos do Xbee, tendo estes um custo mais elevado.

Definindo a função de cada Xbee na rede, é possível projetar a topologia da WSN. A figura 2-6 ilustra diversos tipos de topologias e mostra como devem ser configurados os Xbee para implementar cada uma delas.

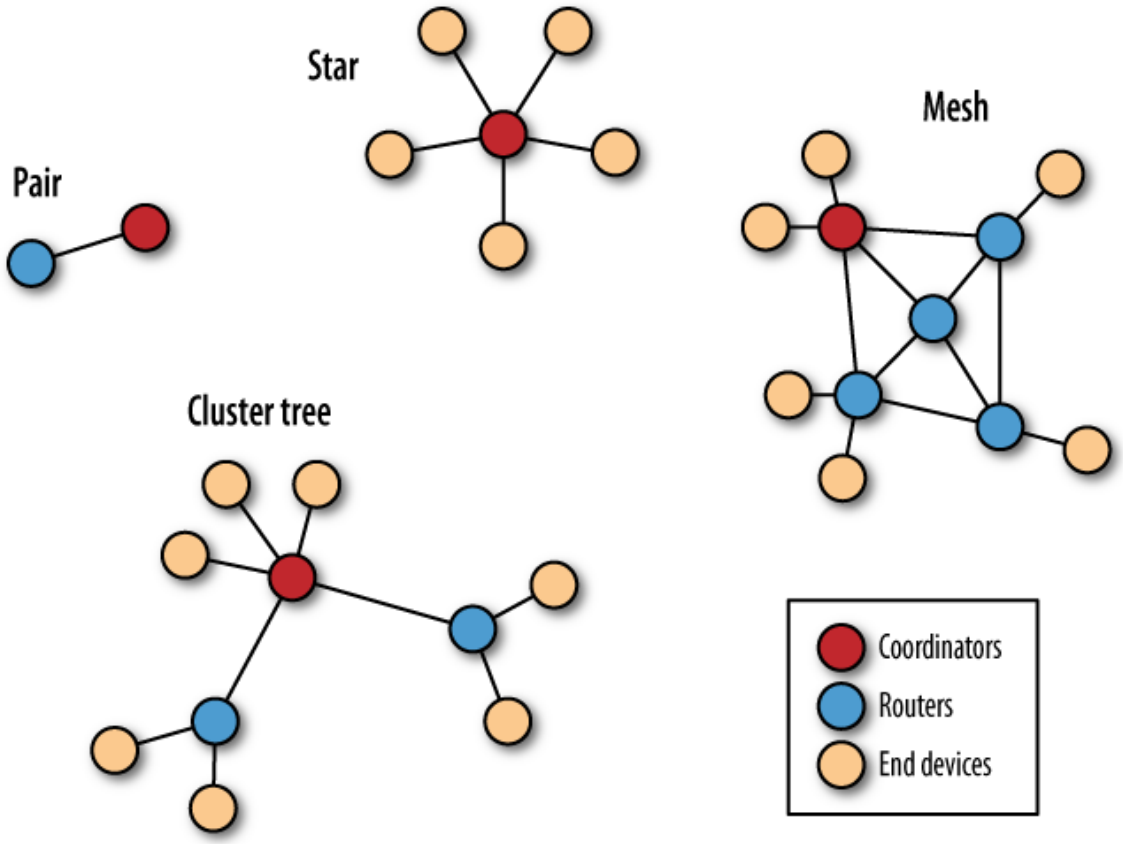
Classificação quanto à WSN a que pertencem:

Ao programar o módulo Xbee deve-se definir o identificador da *Personal Area Network*, PAN ID. Trata-se de um número de 16 bits responsável por “dar nome” à WSN. Todos os nodos de uma mesma WSN devem ter o mesmo PAN ID. Isso permite, por exemplo, que duas WSN operem no mesmo espectro de frequência, no mesmo local, já que os módulos ignoram as mensagens que possuam um PAN ID diferente do seu próprio.

Classificação quanto ao ID do nodo:

Além de um PAN ID, cada módulo precisa ter seu identificador, NODE ID. Esse identificador deve ser único na WSN e é a forma de selecionar um módulo dentro os demais módulos da WSN. Vale ressaltar que dois nodos, em duas WSNs diferentes podem ter o mesmo identificador e ainda assim serem selecionados unicamente devido ao PAN ID.

Figura 2-6: Exemplos de topologias implementadas com módulos Xbee. (FALUDI, 2011)



Existe ainda o endereço MAC do nodo, que é um número de 64 bits que possibilita a identificação unívoca do dispositivo independentemente do PAN ID. Este endereço, porém, é configurado em hardware durante a fabricação do dispositivo e não pode ser alterado por meio do X-CTU.

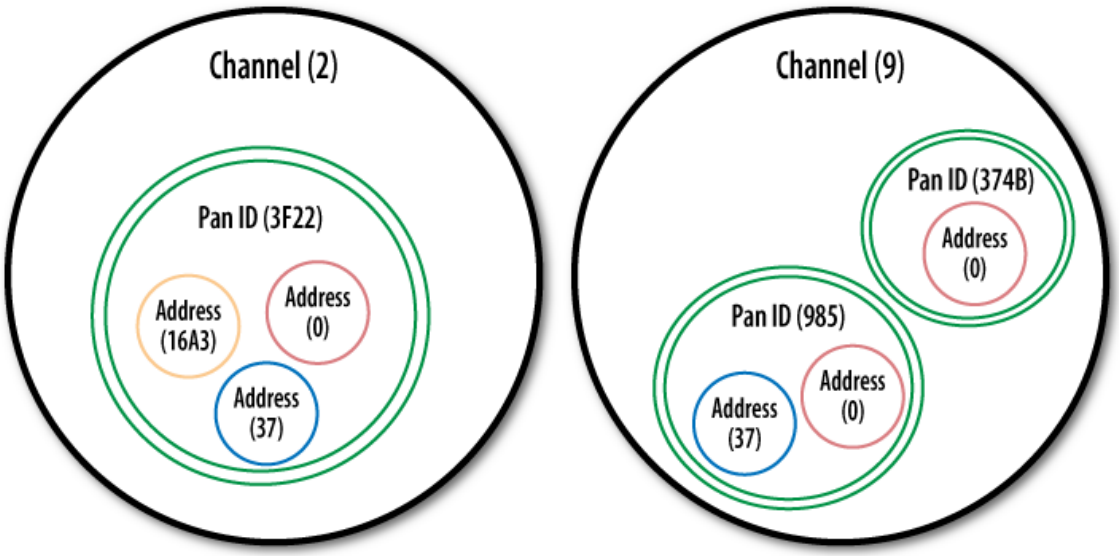
Classificação quanto ao ID do nodo:

Essa última classificação diz respeito ao dispositivo com quem se deseja trocar mensagens. Supondo-se que o nodo A queira enviar mensagens para o nodo B e que o nodo B queira enviar mensagens para o nodo A. Supondo ainda que o nodo A

tenha o NODE ID **0x0001** e que o nodo B tenha o NODE ID **0x0002**. Para que os dois se comuniquem é preciso que o identificador DESTINATION ID do nodo A seja igual ao identificador NODE ID do nodo B e que o identificador DESTINATION ID do nodo B seja igual ao identificador NODE ID do nodo A. Além disso, os PAN ID de ambos devem ser iguais, para que eles façam parte da mesma WSN. É possível ainda configurar um nodo como *broadcaster,* ou seja, que as mensagens deste nodo sejam enviadas a todos os demais nodos de mesmo PAN ID. Vale destacar que a forma de enviar uma mensagem para um subgrupo de nodos dentro da WSN é a realização de um broadcast e do tratamento por software, no micro controlador, por meio da inserção de um cabeçalho de identificação do nodo, da mensagem recebida de acordo com um protocolo a ser desenvolvido pela aplicação. Mais detalhes sobre este tipo de recurso serão dados no capítulo 3, onde será descrita a implementação realizada nos micro controladores.

À primeira vista, a grande quantidade de classificações para um nodo podem levar o desenvolvedor a se confundir. A figura 2-7 ilustra graficamente a funcionalidade dos identificadores descritos.

Figura 2-7: Nodos agrupados de acordo com canal, PAN ID e NODE ID. (FALUDI, 2011)



Na figura 2-7 pode-se observar que dispositivos em canais diferentes estão completamente isolados. Pode-se ver ainda que, mesmo estando num mesmo canal (9), dois dispositivos com o mesmo NODE ID (0) podem coexistir, pois possuem

PAN IDs diferentes (985 e 374B). Porém, dentro de uma mesma PAN, todos os nodos devem ter NODE IDs diferenciados.

Neste projeto, buscando obter a arquitetura de WSN discutida anteriormente e mostrada na figura 2-2, foram configurados quatro nodos com a função de *end-device* e um nodo com a função de *coordinator* formando uma topologia em estrela, como a da figura 2-6 (*Star*). Deve ser feita a ressalva de que, numa topologia desse tipo, todos os nodos sensores devem estar dentro do raio de alcance do *coordinator* (cerca de 30 metros). Uma forma de contornar este problema é a introdução de dispositivos com função de router, para implementar uma topologia clusterizada, como a mostrada na figura 2-6 (*Cluster Tree*). Porém, isso não foi feito por não ser necessário para uma prova de conceito como este projeto de graduação, além de aumentar os custos do hardware. Como mencionado anteriormente, os modelos de Xbee com a funcionalidade de *router* são mais caros.

Uma vez concluídas estas configurações, a topologia da rede está definida e os módulos Xbee prontos para serem integrados com os micro controladores. Isso deve ser feito por meio da interface serial comum do dispositivo, sem que seja necessária qualquer adaptação no software. Um caractere escrito na saída serial pelo micro controlador remetente será lido pelo receptor de forma transparente. O Xbee se encarrega de toda a comunicação.

No capítulo 3 serão descritas as implementações da central e dos nodos sensores, incluindo as interfaces do *Arduino* e do *Raspberry Pi* com os módulos Xbee que os interconectam.

3 SISTEMAS EMBARCADOS

Este capítulo discorre de forma simplificada sobre o funcionamento dos sistemas embarcados *Raspberry Pi* e Arduino que foram utilizados para a criação do projeto. Ademais, são tecidas considerações sobre as implementações realizadas em cada um dos sistemas embarcados para que fosse possível a implantação do sistema.

3.1 DESCRIÇÃO E SELEÇÃO DOS SISTEMAS EMBARCADOS

O componente mais importante de um nodo em uma WSN é o elemento de processamento, ou seja, o elemento responsável pela inteligência do sistema. Na implementação da WSN deste projeto foram usados dois tipos de elementos de processamento.

No dispositivo central desta arquitetura ocorre o processamento e a disponibilização dos dados enviados pelos demais dispositivos. Além disso, o dispositivo central deve possuir interfaces de rede (requisito de *hardware*) e implementar um servidor WEB (*software*) para que os dados adquiridos sejam acessíveis por meio da Internet. Esse conjunto de funcionalidades é mais sofisticado que as funcionalidades típicas exigidas de um nodo de uma WSN. Por isso, é necessário um hardware mais potente. Esse conjunto de requisitos justifica a escolha do *Raspberry Pi* como elemento de processamento na central da WSN implementada. O *Raspberry Pi* e o conjunto de funcionalidades nele implementadas serão descritos a seguir na seção 3.2.

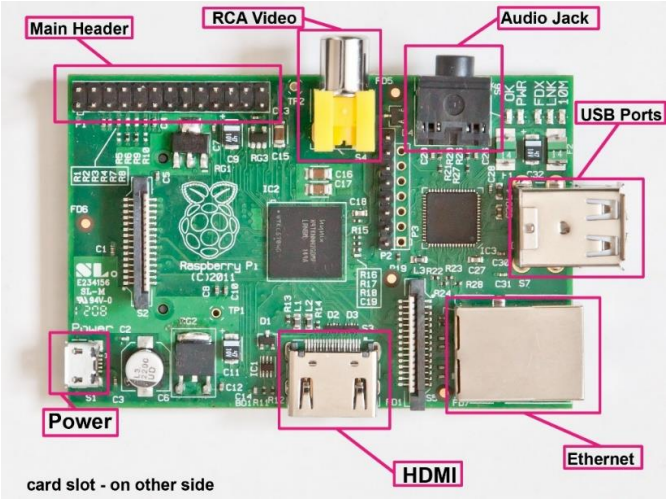
Os demais nodos da WSN implementada dever ser capazes de realizar comunicação serial com os módulos Xbee a eles conectados e executar funções típicas de micro controladores como: conversão Analógico/Digital (A/D), programação de interrupções por tempo e output de valores lógicos por meio de GPIO (*General Purpose Input Output*). O dispositivo escolhido para realizar estas funções foi o Arduino, devido a sua robustez, boa documentação e facilidade de

utilização. O sistema embarcado Arduino e a implementação nele realizada serão descritos na seção 3.3.

3.2 *RASPBERRY PI* E A CENTRAL DA WSN

O *Raspberry Pi* é um projeto da *Raspberry Pi Foundation,* organização situada no Reino Unido, que tem por objetivo popularização do ensino de programação para crianças e adolescentes. A partir de seu lançamento, o *Raspberry Pi* rapidamente passou a ser conhecido com o menor e mais barato computador pessoal do mundo. Além do baixo custo ($25 na época do lançamento) e tamanho reduzido, possui as vantagens que possuir todas interfaces básicas de um PC comum: vídeo HDMI, áudio, Ethernet, USB, entre outros. Além disso, conta com um conjunto de interfaces GPIO, que facilitam a utilização da solução em aplicações de Hardware embarcado. O armazenamento dos dados persistentes é realizado em um cartão SD, uma alternativa de tamanho reduzido (mantendo a portabilidade do *hardware*), fácil acesso e alta velocidade de leitura e escrita. As interfaces básicas do *hardware* do *Raspberry Pi* são identificados na figura 3-1.

Figura 3-1: Componentes do *hardware* do *Raspberry Pi*.



Uma importante vantagem do *Raspberry Pi* está no fato de que nele pode ser executado um sistema operacional baseado em Linux, permitindo a utilização de diversas funcionalidades e programas existentes nesse sistema operacional, já validados e amplamente documentados. Além disso, Linux é um *software* livre que pode ser utilizado sem criação de custos adicionais ao sistema. Atualmente existem algumas variações de distribuições do Linux especificamente otimizadas para o *Raspberry Pi*, entre elas a mais amplamente utilizada, mais documentada e recomendada pela *Raspberry Pi Foundation* é a distribuição Raspbian, que foi utilizada neste projeto.

Somada a todas as vantagens supracitadas existe ainda a questão do conforto do usuário do sistema quando se trata da instalação. Primeiramente, o Raspberry Pi ocupa muito pouco espaço no ambiente em que for ser instalado. A figura 3-2 ilustra o tamanho da central do sistema, que cabe na palma de uma mão. O Linux conta ainda com uma forma de inicializar o *software* desenvolvido para a central da WSN no momento da inicialização do sistema operacional, tornando a central *plug-and-play*.

Figura 3-2: Central da WSN implementada com o *Raspberry Pi*.



A instalação do sistema Raspbian no Raspberry Pi é amplamente documentada e será omitida deste texto, podendo ser encontrada, por exemplo, no *site* oficial do *Raspberry Pi*. Serão aqui abordados o procedimento para implementação do servidor Web (LAMP) no *Raspberry Pi* e o *software* desenvolvido para a aquisição dos dados recebidos pela interface USB conectada ao Xbee e disponibilização dos dados no servidor WEB.

Instalação e Configuração do LAMP Server e Setup da Rede

Um *LAMP server* é uma das configurações mais comuns de servidores Web. Ele consiste de um computador rodando o sistema operacional Linux, o servidor Web HTTP Apache, o servidor de banco de dados MySQL e a linguagem de programação PHP. Com esta configuração básica é possível hospedar a grande maioria dos sites encontrados na Internet.

Neste trabalho, um dos objetivos é permitir o acesso aos dados de consumo de energia do domicílio por meio de um *Web Browser*. Por isso, é necessário que o *Raspberry Pi* torne-se um servidor Web. Como o *Raspberry Pi* é um dispositivo que

roda sistema operacional Linux, é natural que se utilize um servidor *LAMP*. Para instalar o Apache, o MySQL e o PHP no Linux são utilizados os seguintes comando, como super-usuário:

sudo apt-get install apache2

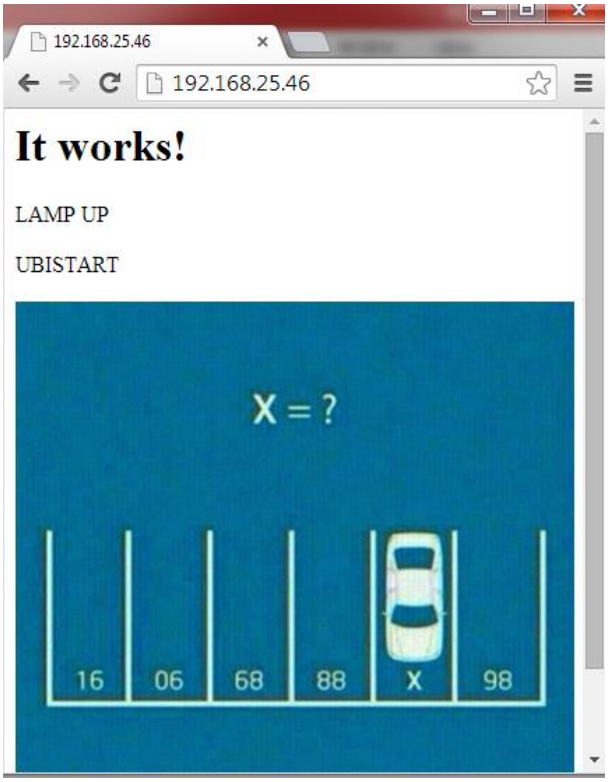
sudo apt-get install mysql-server

sudo apt-get install php5

sudo apt-get install php5-mysql

Com isso, será criado no *Raspberry Pi* o diretório **"/var/www"**. Os arquivos presentes neste diretório poderão ser acessados por um navegador, bastando digitar, estando numa mesma rede local, o IP *do Raspberry Pi*. A figura 3-3 mostra o acesso a um site hospedado no *Raspberry Pi*.

Figura 3-3: Acesso a site hospedado no *Raspberry Pi*.



Como a maioria dos domicílios não conta com um IP público fixo para cada interface de rede, utiliza-se normalmente o protocolo NAT. Por isso, para permitir que

usuários externos à LAN em que se encontra servidor Web o acessem, é necessário configurar a tabela de encaminhamento do Roteador, direcionando os pacotes que chegam pela porta 80 (*HTTP*) para o IP do *Raspberry Pi*, como mostra a figura 3-4.

Com isso está completada a configuração do servidor Web que será utilizado para acessar os dados de consumo de energia. Opcionalmente, pode-se reduzir a memória destinada à parte gráfica do sistema operacional, já que a central da WSN não será utilizada diretamente pelos usuários, funcionando apenas como provedor de serviços.

Figura 3-4: Configuração da tabela de encaminhamento do roteador.



Há um último problema que deve ser solucionado para que o servidor Web funcione de forma adequada. A maioria das companhias provedoras de acesso à Internet destina a seus usuários comuns um IP dinâmico. Este IP muda frequentemente, muitas vezes mais de uma vez por dia. Isso dificulta o acesso ao servidor Web por parte de usuários de fora da LAN. Para contornar este problema foi utilizado um DDNS.

O DDNS é um tipo especial de servidor de DNS que funciona de forma a contornar a mudança constante de IP de um ponto de acesso. Toda vez que o IP do ponto de acesso muda, o roteador envia uma notificação para o servidor de DDNS. O servidor de DDNS por sua vez cuida de atualizar as tabelas de DNS, permitindo que a tradução de nome para IP ocorra sempre com o IP correto, notificado pelo roteador do domicílio (THOMSON; REKHTER; BOUND, 1997). Existem diversos provedores de serviço de DDNS, muitos deles gratuitos. Neste projeto foi utilizado o NoIP.

Concluídas estas etapas o servidor Web está configurado e pronto para ser utilizado.

Software de Aquisição e Disponibilização dos Dados

Além de funcionar como servidor Web, a central do sistema tem mais duas funções fundamentais: a aquisição dos dados oriundos dos sensores e o atendimento das requisições, que podem ser de acionamento de dispositivo ou de relatório de consumo de energia. Para cumprir estas tarefas foi implementado um software na linguagem de programação Java.

Os módulos Xbee são conectados no *Raspberry Pi* pela interface USB. Como descrito anteriormente, a comunicação entre os módulos Xbee é transparente para os dispositivos aos quais eles se encontram conectados. Dessa forma, se um nodo sensor envia o caracter 'a' por sua interface serial, que está conectada ao Xbee, a central receberá o caracter 'a' em sua interface USB, desde que os Xbees estejam devidamente configurados como descrito no Capítulo 2. Por esta razão, para enviar e receber dados, basta que o código Java escreva e leia da interface serial correspondente ao USB. No Linux, esta interface pode ser acessada por meio de um arquivo que é montado no diretório "/dev" quando um dispositivo USB é conectado. Normalmente esses arquivos são nomeados "ttyUSB0", "ttyUSB1" e assim por diante.

Para trabalhar com a interface serial USB foi utilizada a biblioteca RXTXComm. A biblioteca RXTXComm implementa um conjunto de0 métodos para iniciar e manejar uma comunicação serial. Entre eles os métodos providos pela biblioteca, os mais importantes, que são utilizados neste trabalho são: abrir uma conexão com uma determinada porta serial, escrever dados no buffer de envio, enviar os dados do buffer e ler dados recebidos no buffer de recebimento.

Para abrir a conexão com uma determinada porta serial é necessário saber a localização e o nome do arquivo correspondente a esta porta no sistema operacional. Como mencionado anteriormente, no caso do *Raspberry Pi* a conexão é montada no endereço **"/dev/ttyUSB0"**. A sequência de métodos para iniciar a conexão com o dispositivo será explicada em detalhes.

Primeiramente é necessário obter o ID da porta serial de destino a partir da *String* relativa a sua localização no sistema operacional. A figura 3-5 mostra como realizar este procedimento.

No código da Figura 3-5, é implementado um *loop* que percorre a lista de dispositivos seriais conectados. Quando é encontrado o dispositivo cujo nome é o mesmo do definido na variável ***portName***, neste caso "/dev/ttyUSB0" atualiza-se a variável ***portId*** com o ID correspondente a este dispositivo. O ID armazenado em ***portId*** será utilizado para abrir a conexão serial em seguida. Caso não seja encontrado um dispositivo com o nome igual ao armazenado na variável ***portName***, é mostrada a mensagem de erro: "Dispositivo não encontrado".

Figura 3-5: Obtenção do ID da porta serial a ser utilizada.

```
CommPortIdentifier portId = null;

String portName = "/dev/ttyUSB0"

Enumeration portEnum = CommPortIdentifier.getPortIdentifiers();

while (portEnum.hasMoreElements()) {
    CommPortIdentifier currPortId = (CommPortIdentifier) portEnum.nextElement();
    if (currPortId.getName().equals(portName)) {
        portId = currPortId;
        break;
    }
}

if(portId == null){
    System.out.println("Dispositivo nao encontrado");
}
```

De posse do ID da porta serial, é possível abrir a conexão serial com o dispositivo desejado, como mostrado na figura 3-6.

Figura 3-6: Abertura da conexão serial.

```
try {
        //abertura da porta serial.
        serialPort = (SerialPort) portId.open(this.getClass().getName(),TIME_OUT);

        //configuracao dos parametros da porta
        serialPort.setSerialPortParams(DATA_RATE,
                SerialPort.DATABITS_8,
                SerialPort.STOPBITS_1,
                SerialPort.PARITY_NONE);

        //Abertura dos streams
        BufferedReader input = new BufferedReader(new InputStreamReader(serialPort.getInputStream()));
        OutputStream output = serialPort.getOutputStream();

        //Cria o event listener
        serialPort.addEventListener(this);
        serialPort.notifyOnDataAvailable(true);
} catch (Exception e) {
    System.err.println(e.toString());
}
```

Primeiramente é chamado o método ***open*** sobre a instancia ***portId***, que contém o ID da porta seria a ser aberta. O método ***open*** recebe como parâmetros uma String que será a identificação da aplicação que está abrindo a porta serial um valor de *timeout* da conexão com a porta, em milissegundos, a partir do qual a conexão é encerrada.

Em seguida, continuando o processo de inicialização da conexão, devem realizadas as configurações da porta, entre elas a velocidade da comunicação, em bauds, número de bits de dados, número de stop bits e opção ou não por bit de paridade. É importante que todos os dispositivos que desejam se comunicar tenham os mesmos parâmetros de configuração. Caso contrário a comunicação não funcionaria. No caso deste trabalho, os Arduinos devem ter as mesmas configurações aqui atribuídas para a central.

Definidas as configurações, os *streams* de dados são atribuídos a *buffers* de entrada e saída. Por fim é adicionado um *Event Listener,* que dispara o tratador de eventos quando há dados disponíveis nos buffers associados à porta serial em questão. Isso finaliza o processo de pré-configuração das interfaces com o USB. Após estes passos a interface USB está pronta para ser utilizada.

Como foi definido um *Event Listener* que avisa quando há dados disponíveis no *buffer,* basta implementar o método tratador do evento para manipular os dados de acordo com a necessidade da aplicação. O método a ser implementado é o

serialEvent. A figura 3-7 mostra um exemplo de implementação do método ***serialEvent*** que apresenta no *console* os dados recebido pela interface USB.

Figura 3-7: Recepção dos dados com o método *serialEvent*.

```
public synchronized void serialEvent(SerialPortEvent oEvent) {
    if (oEvent.getEventType() == SerialPortEvent.DATA_AVAILABLE) {
        try {
            String inputLine=input.readLine();
            System.out.println(inputLine);
        } catch (Exception e) {
            System.err.println(e.toString());
        }
    }
}
```

A central de controle da WSN deste projeto realiza dois tipos de comunicação via interface USB: envio de comando de ligar/desligar um equipamento e recepção dos dados de consumo de energia dos diversos aparelhos. Para possibilitar a implementação simultânea de ambas as funcionalidades foram utilizada duas *threads*. Uma delas fica sempre escutando a porta USB e lendo os dados de consumo e a segunda fica esperando comandos do usuário e transmitindo estes comandos para o dispositivo que se deseja ligar/desligar.

Com isso está finalizada a implementação da comunicação da central da WSN com os nodos sensores. Resta ainda criar uma forma dos usuários do sistema interagirem com a central. A alternativa utilizada para implementar essa interação foi a criação de um Web Service no *software* Java da central.

O funcionamento básico de um Web Service consiste da associação de um método do código Java a uma determinada requisição HTTP. Por exemplo, um usuário do sistema que se encontra em seu escritório deseja desligar o ar condicionado de sua casa, que ele esqueceu ligado ao sair para trabalhar. Para isso, basta que ele abra seu *Web Browser* e entre com a seguinte URL:

siteladecasa.noip.me:8080/module/ArCondicionado2/turnOff

Na URL, **"siteladecasa.noip.me"** é o endereço cadastrado no servidor de DDNS. A porta 8080 é a porta associada ao Web Service e está direcionada no roteador da casa para o IP da Central, de acordo com o mostrado anteriormente na figura 3-4. Se o sufixo **"module/ArCondicionado2/turnOff"** estiver associado a uma instância

no Web Service, o método ***turnOff*** será invocado, para a instância **ArCondicionado2**, enviando o comando de desligar o equipamento pela porta serial.

Embora seja possível utilizar o sistema diretamente a partir das URLs, o objetivo de utilizar o Web Service é a facilidade de associar um botão em um site ou em um aplicativo para *mobile* a uma URL. Portanto, não se espera que o usuário utilize diretamente as URLs. Posteriormente no capítulo 4 é mostrado um exemplo de interface para interação com o sistema que torna o uso das URLs transparente para o usuário.

A porta serial do *Raspberry Pi* conectada a um módulo Xbee, que está previamente configurado de acordo com o descrito no capítulo 2, envia o comando para o Xbee do nodo sensor conectado ao aparelho de ar condicionado. No nodo sensor o Arduino cuida de interpretar a mensagem e desligar o equipamento. Resta agora descrever como o Arduino realiza esta tarefa.

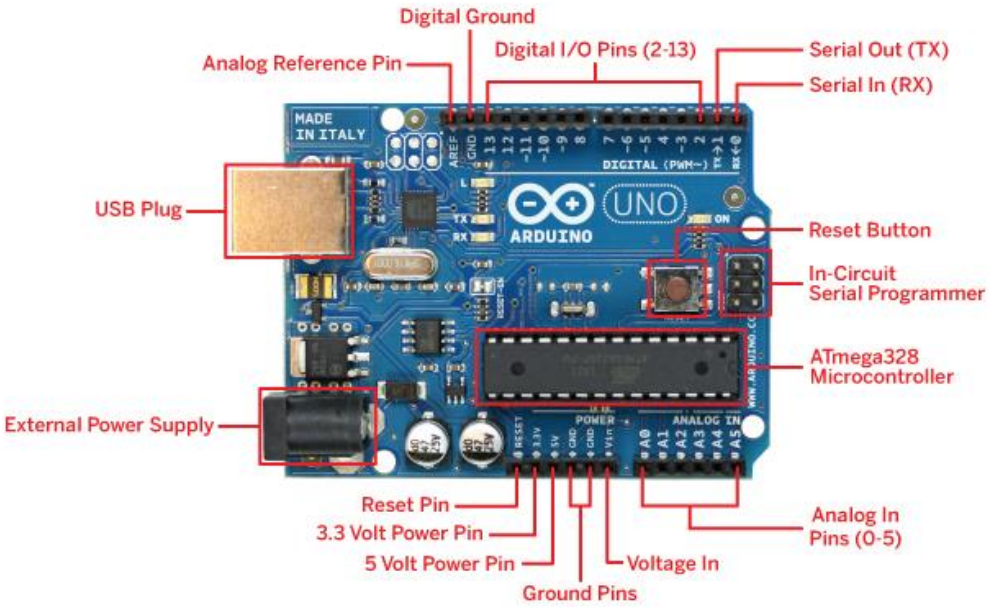
3.3 ARDUINO E OS NODOS SENSORES

Nodos sensores de uma WSN são dispositivos dotados de capacidade de sensoriamento, armazenamento, processamento e comunicação *wireless* (BERNDT, 2006). Para implementar esse conjunto de funções foi utilizado o Arduino.

O Arduino é uma plataforma de prototipação de código e hardware abertos que tem por objetivo propiciar um ambiente de fácil utilização para criação de protótipos de sistemas embarcados variados. Existem diversos modelos de Arduino que variam quanto ao tamanho, preço e recursos de hardware disponíveis. Os nodos sensores da WSN implementada neste projeto utilizam o Arduino UNO, que é o modelo mais clássico e comumente encontrado. Não obstante, poderiam ter sido utilizados modelos como o Arduino Nano ou Micro, que tem um tamanho mais reduzido. Isso não foi feito devido à não disponibilidade das peças.

Dentre os recursos que facilitam a utilização do Arduino destacam-se a presença de um regulador de tensão já embarcado, dispensando componentes eletrônicos adicionais para ligar o micro controlador, e a presença de *pin-heads* ligados aos barramentos do micro controlador facilitando a ligação do Arduino a outros componentes. A figura 3-8 mostra as principais interfaces do Arduino.

Figura 3-8: Arduino UNO e suas interfaces.



O processamento do Arduino UNO é realizado em um processador Atmel ATmega328, com *clock* de 16 MHz, que provê diversos recursos de *hardware.* Dentre eles os seguintes serão fundamentais ao projeto dos nodos sensores:

- Interrupções por tempo.

- Interrupções de *hardware*.

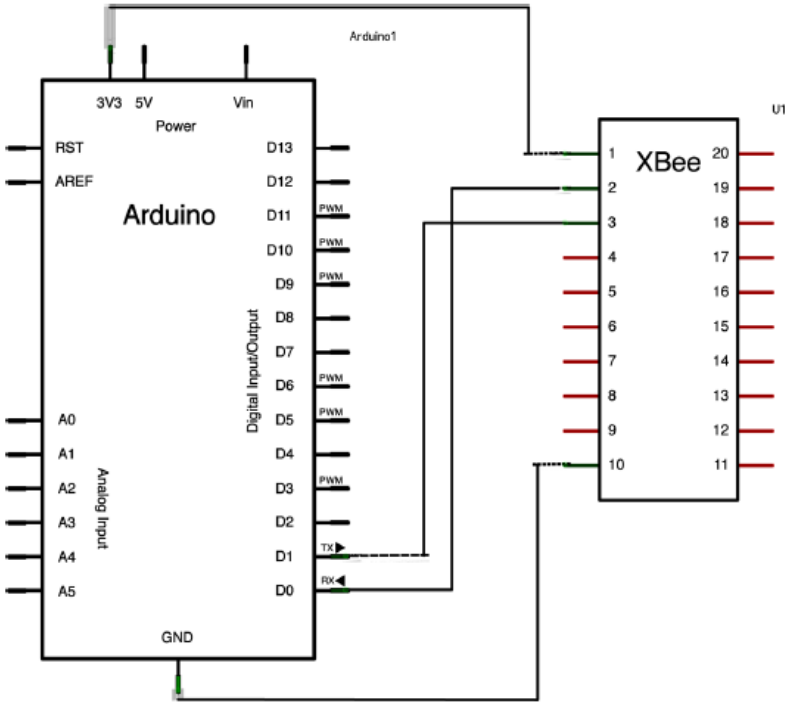
- Conversão Analógico-Digital para leitura de sensores.

- Comunicação serial via USART.

Para o correto funcionamento do nodo sensor é necessário realizar a interface do Arduino com três *hardwares* adicionais: o Xbee, o sensor de corrente e o circuito de acionamento de equipamentos.

A interface do Arduino com o Xbee é simples. Basta alimentar o Xbee com a saída de 3.3V do arduino, interconectar os pinos de *ground* e ligar o pino ***tx*** do Arduino com o pino ***rx*** do Xbee e o pino ***rx*** do Arduino com o pino ***tx*** do Xbee. A figura 3-9 mostra o esquemático dessa conexão.

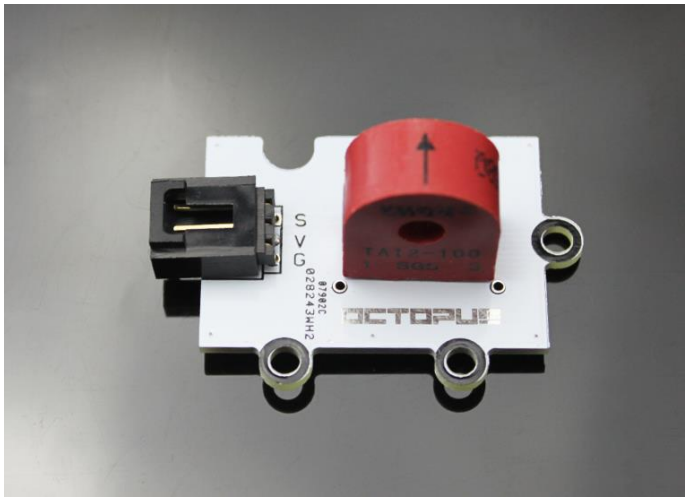
Figura 3-9: Interface do Arduino com o Xbee.



Para enviar e receber comandos do Xbee, basta configurar a interface serial do Arduino na mesma velocidade da configurada na central da WSN (9600 bauds), utilizando a função ***Serial.begin***, e utilizar as funções ***Serial.print*** para enviar dados e ***Serial.read*** para recebê-los.

A segunda conexão de *hardware* necessária é a do sensor de corrente. O sensor de corrente utilizado é o TA12-100, mostrado na figura 3-10, funciona por indução eletromagnética, portanto o fio do aparelho cuja corrente se deseja medir deve passar dentro da bobina do sensor (em vermelho na figura 3-10).

Figura 3-10: Sensor de corrente TA12-100.

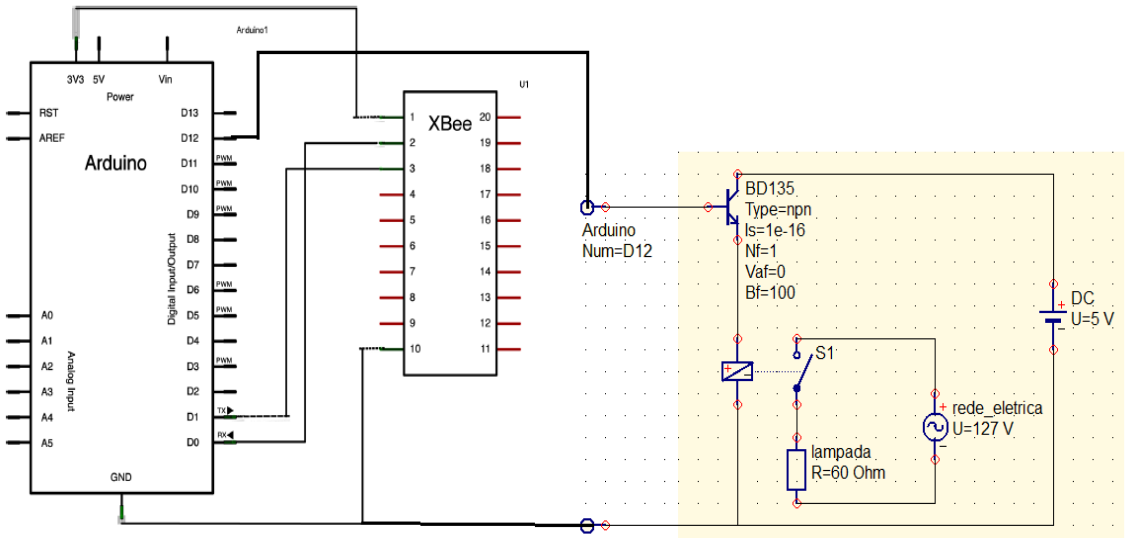


O sensor deve ser alimentado com 5 V e retorna uma tensão de 0 a 5 V proporcional à corrente medida, que pode ser de 0 a 5 A. O valor de tensão retornado pelo sensor deve ser ligado em uma das portas do Arduino com capacidade de conversão analógico-digital (pinos A0 até A5).

No *software* embarcado do Arduino, quando for feita a chamada de função ***analogRead***, passando como parâmetro o número do pino em que o sensor de corrente está conectado, o valor de corrente medido pelo sensor no momento da chamada da função será convertido para um valor digital correspondente a um inteiro de 0 a 1023, proporcional ao valor de 0 a 5 V fornecido pelo sensor.

A terceira interface de *hardware* consiste do circuito para acionamento dos equipamentos a serem controlados e monitorados. Este circuito é um pouco mais complexo pois faz adaptação entre um sistema digital de baixa potência, o Arduino, e a rede elétrica, onde estará conectado o aparelho. Para ligar e desligar o equipamento ligado à rede elétrica deve ser utilizado um relé de 5VDC, ou seja, que chaveia quando alimentado com 5V ou mais. Apesar do nível lógico alto do arduino ser 5 V, os pinos de saída não têm capacidade de fornecer corrente suficiente para acionar o relé. Isso fazer necessária a introdução de um transistor com a funcionalidade de *driver* de corrente. Assim o pino do Arduino que será responsável pelo chaveamento fica ligado na base do transistor, que funciona como chave. A figura 3-11 mostra o esquema de ligação do circuito de acionamento.

Figura 3-11: Circuito de acionamento ligado ao Arduino.



De acordo com o esquema da figura 3-11 o pino D12 do Arduino é ligado na base do transistor. Quando o pino D12 está no nível lógico alto, ele satura o transistor ativando o relé e ligando a lâmpada (ou outro equipamento qualquer). Quando o pino D12 está no nível lógico baixo o transistor passa a estar cortado, desligando o relé e o equipamento.

Estando montado este circuito, o acionamento/desligamento do equipamento passa a consistir de alterar o valor lógico do pino D12. Isso é realizado por meio da função: ***"digitalWrite(12 , nivel_logico)"***. Na chamada dessa função ***"nivel_logico"*** pode assumir os valores ***"HIGH"*** ou ***"LOW"***. A decisão de com qual nível lógico deve ser configurada a interface de saída do pino D12 é baseada nos comando recebidos pela interface serial do Arduino, que está conectada ao módulo Xbee, recebendo comandos da central da WSN, ou seja, do *Raspberry Pi*.

Da perspectiva do software embarcado existem três funcionalidades básicas que o nodo sensor deve executar. São elas:

1- Receber comandos da central.

2- Ligar/Desligar equipamentos de acordo com o comando recebido.

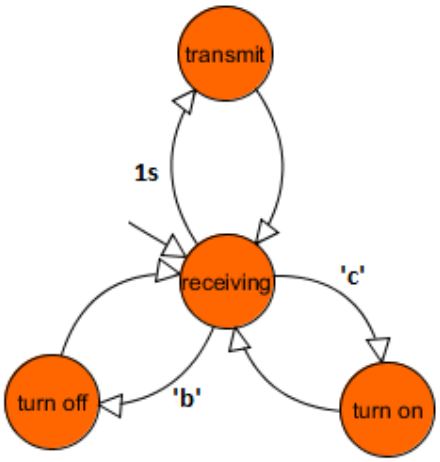
3- Medir a corrente, calcular e enviar o consumo do equipamento para a central.

Como a central de controle sempre se comunica com os nodos por *broadcast*, é necessário criar um mecanismo para identificar para qual nodo se destina uma determinada mensagem e ainda, de qual nodo está chegando uma determinado dado de consumo. Para realizar isso, cada mensagem passa a ser acompanhada por um byte identificador do nodo. Com isso as mensagens têm a seguinte forma.

Enviado pela central	**ID NODO DESTINATÁRIO** 1 byte ('a','b','c'...)	**COMANDO LIGAR/DESLIGAR** 1 byte ('c' ou 'b')
Enviado pelos nodos	**ID NODO REMETENTE** 1 byte ('a','b','c'...)	**VALOR DE CONSUMO** 4 bytes (VALOR)

Um questão central quando se trata de sistemas embarcados é o compartilhamento do tempo de processamento, já que na maioria das vezes, incluindo o caso do Arduino, não existe um sistema operacional para cuidar do escalonamento das tarefas. O livro *"Programming Embedded Systems: An Introduction to Time-Oriented Programming"* (VAHID; GIVARGIS; MILLER, 2012) define uma série de técnicas de programação orientada a tempo compartilhado. Segundo os autores, quando há um grande número de tarefas a ser desempenhada pelo processador passa a ser necessária a implementação de uma máquina de estados que compartilhe o tempo de processamento entre estas tarefas. Para compartilhar o tempo de processamento entre as tarefas 1, 2 e 3 supracitadas foi definida a máquina de estados mostrada na figura 3.12.

Figura 3-12: Máquina de estados implementada no Software do Arduino.



De acordo com a máquina de estados mostrada na figura 3-12, o *software* embarcado inicia-se no estado ***receiving***, que é o estado ocioso do sistema. Ele pode sair do estado ***receiving*** em três ocasiões. Quando recebe o caracter **'c'**, advindo da central de comando, por meio do xbee, o sistema entra no estado ***turn on***, onde o equipamento é ligado modificando-se o valor da saída do pino D12. Após ligado o equipamento, o sistema volta ao estado ***receiving*** e aguarda outro comando. O estado ***turn off*** funciona da mesma forma, com a diferença que o caracter recebido da central muda de **'c'** para **'b'**.

Tanto a transição para o estado ***transmitting*** como as tarefas executadas nesse estado são mais complexas que as tarefas dos demais estados. Os nodos sensores devem reportar de segundo em segundo o consumo de energia do equipamento da central. Por isso, a cada segundo o nodo deve entrar no estado ***transmitting***. Para isso, é usada uma interrupção por tempo. A configuração da interrupção por tempo é mostrada na figura 3-13.

Figura 3-13: Configuração da interrupção por tempo no Arduino.

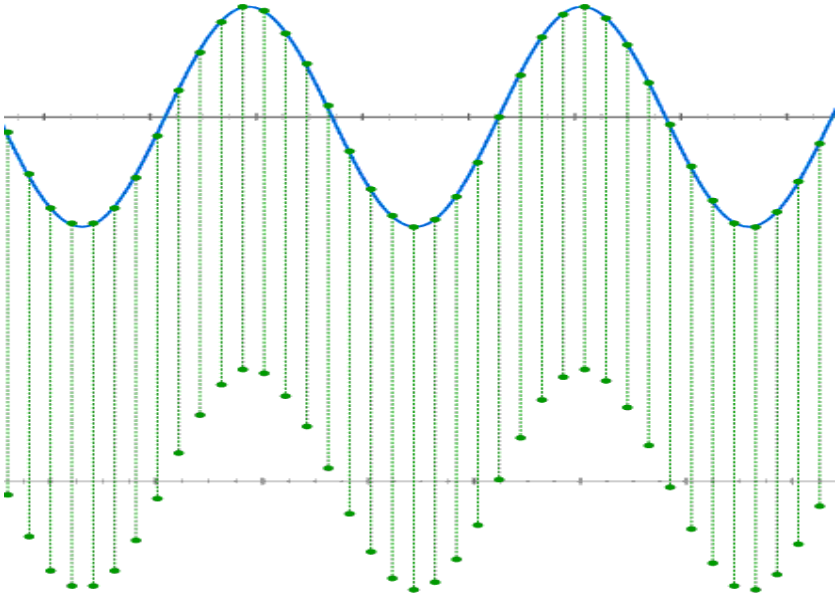
```
//Interruptions
        Timer1.initialize(1000);
        Timer1.attachInterrupt(readSensor);
}

void readSensor(){
        state = transmitting;
}
```

Primeiramente é definido o valor de um *timer* em milissegundos. Este *timer* é um contador que é implementado em *hardware*. O contador incrementa um *buffer* periodicamente. Quando ocorre o *overflow* deste *buffer* o processador recebe um sinal e interrompe a execução do software para executar a rotina de tratamento de interrupção definida no momento da criação da interrupção. Neste caso a rotina de tratamento faz com que a máquina de estados entre no estado ***transmitting***. Como o contador é implementado em hardware, o processador não fica ocupado durante a contagem, podendo executar as demais funções ao mesmo tempo.

O estado ***transmitting*** deve realizar a medição do consumo de energia e enviar o valor medido para a central da WSN. Porém, medir o consumo de energia não é simples já que para calcular a potência de um sinal senoidal é preciso obter o valor de pico da corrente, ou seja, o maior valor que ela adquire dentro do período de oscilação da onda. Para obter esse valor é preciso amostrar a onda em um período de amostragem consideravelmente menor que o período da senóide da rede elétrica, que é de 16,66 milissegundos. Nessa implementação são feitas 20 amostragens por período, ou seja, é retirada uma amostra a cada 0,83 ms, permitindo um erro percentual inferior a 5%. Considerando o maior valor obtido entre as amostras pode se obter um valor aproximadamente igual ao valor máximo de corrente. Como explicado anteriormente, a amostra é obtida por meio da chama de função ***analogRead***, passando como parâmetro o pino ao qual o sensor de corrente está conectado, neste caso o pino A0. A figura 3-14 ilustra o processo de amostragem da onda senoidal.

Figura 3-14: Onda senoidal amostrada.



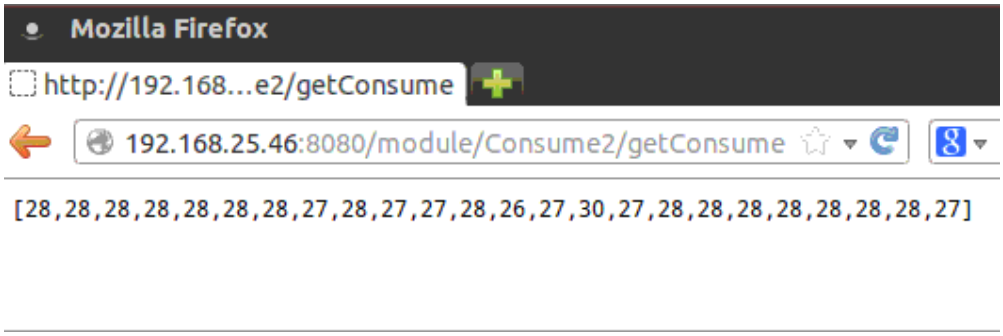
Tendo a corrente de pico do aparelho é possível calcular o seu consumo de energia já que a potência elétrica de consumida por um circuito puramente resistivo é dada por:

$$P = (Vp * Ip) / 2$$

Onde a tensão de pico é nominal da rede elétrica e igual a aproximadamente 180 V. O cálculo exato da potência consumida por cargas de natureza indutiva envolve a medição do ângulo de desfasamento entre a onda de tensão e a onda de corrente. Esse tipo de característica aumenta a complexidade da medição e não será tratada neste projeto, estando relacionada na lista de trabalhos futuros.

Após a realização do cômputo da potência o valor obtido é enviado para a central por meio da chamada de função ***Serial.print***, passando como parâmetro o valor a ser enviado. Ao chegar na central o dado é inserido em um vetor sequencial correspondente ao nodo que o enviou. O vetor pode ser obtido via requisição HTTP da mesma forma que o acionamento/desligamento dos equipamentos, facilitando a criação de interfaces de visualização dos dados. A figura 3-15 mostra o vetor de medições obtido por meio de uma requisição HTTP.

Figura 3-15: Vetor de medições obtido por requisição HTTP.



Dessa forma ficam definidos os funcionamentos dos nodos sensores e da central da WSN. No capítulo 4 será demonstrada a aplicabilidade do sistema por meio da construção de um cenário de testes.

4 IMPLANTAÇÃO E TESTES

Este capítulo cuida da implantação do cenário de testes da WSN, incluindo a criação dos ambientes de interface com o usuário, isto é, um site e um aplicativo para dispositivos móveis.

4.1 DESCRIÇÃO

A proposta deste trabalho é prover uma interface na qual uma pessoa possa monitorar e controlar o consumo de energia dos aparelhos eletrônicos. Para isso é fundamental que seja desenvolvido um ambiente amigável de acesso ao sistema. Como ele se conecta ao usuário pela interface Ethernet do *Raspberry Pi* esse acesso pode ser feito tanto pela rede local, como de qualquer lugar do mundo pela Internet. Portando o sistema deve funcionar nesses dois cenários de teste.

A utilização do Web Service foi escolhida por tornar a implementação de interfaces independente da central da WSN. A central apenas responde as requisições HTTP enviando comando para os nodos da WSN ou retornando para o requisitante os dados de consumo de energia. Isso ajuda também a diminuir a sobrecarga de processamento da central.

Para validar o sistema será implantada uma rede de testes para controle de equipamentos simples característicos de residências. Além disso, será implementado um exemplo simples de interface com o usuário para dispositivos móveis e um *site* que mostra o consumo de energia dos aparelhos em tempo real.

4.2 REDE IMPLANTADA

A rede implantada para realização dos testes consiste da central de monitoramento e controle mais dois nodos sensores. O primeiro monitora uma lâmpada incandescente e o segundo um ventilador. Ambos os nodos realizam a medição do

consumo e o acionamento/desligamento dos equipamentos, embora as duas funções sejam independentes, podendo existir nodos que realizam apenas acionamento/desligamento ou apenas medição do consumo de energia.

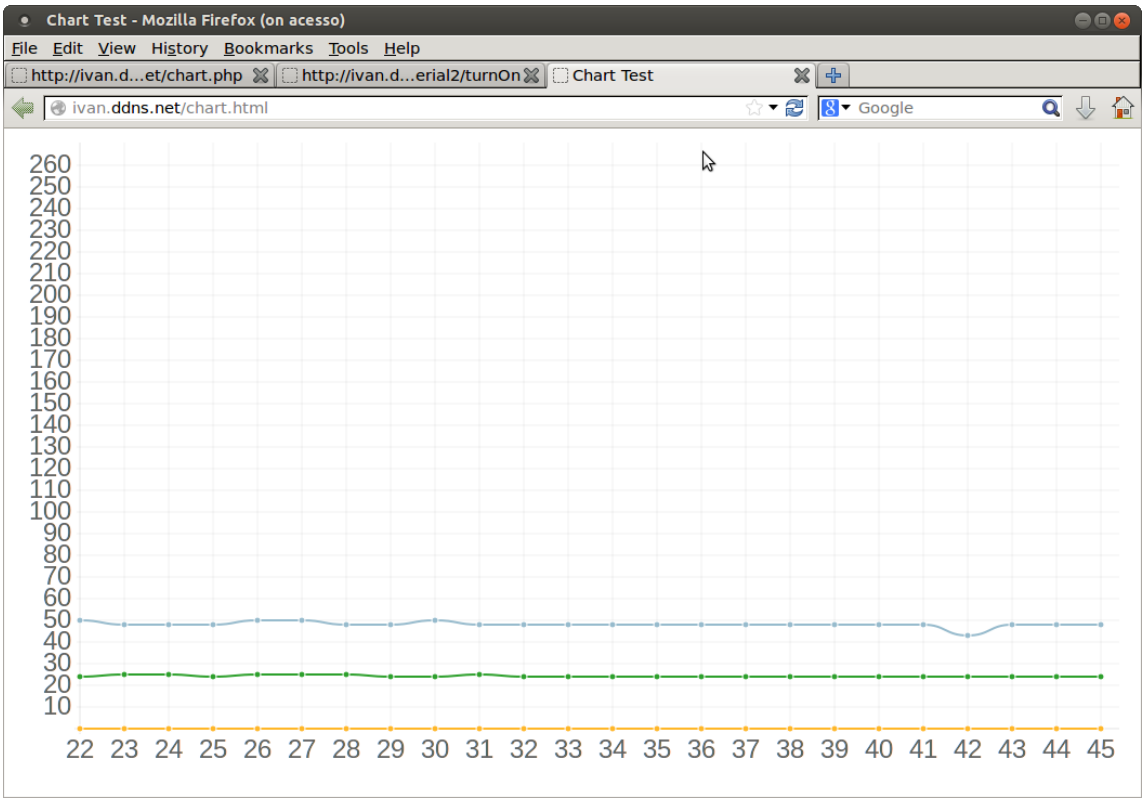
Na central da WSN são mantidos dois vetores de dados de consumo de energia, um para cada nodo sensor. Também são criadas duas instâncias de módulos no Web Service, para permitir ligar/desligar cada um dos módulos individualmente por meio dos métodos ***turnOn*** e ***turnOff***.

A cadastro de um nodo na central é feito por meio de um arquivo de configuração XML, que é lido quando o software Java da central é iniciado. No XML é também definido um caracter identificador do nodo, este caracter é enviado junto a todas as mensagens e serve para identificar qual dos nodos é o destinatário da mensagem. Voltando ao capítulo 2, vale recordar que o módulo Xbee da central é configurado para realizar o *broadcast* das mensagens por ele enviadas. Por isso, todos os nodos recebem todas as mensagens enviadas pela central, identificando as mensagens a eles destinadas e ignorando as demais, de acordo com o identificador. O mesmo identificador é utilizado para separar os dados de consumo de energia entre os vetores de consumo, referentes a cada um dos nodos. Em resumo o identificador serve para diferenciar os nodos.

4.3 INTERFACE COM O USUÁRIO

A primeira interface é simples e consiste de um site que mostra o consumo de energia dos aparelhos em um gráfico. O eixo das abcissas mostra o tempo e o das ordenadas o consumo de cada aparelho em Watts. As curvas de consumo de cada aparelho têm cores diferentes o que facilita a identificação. A interface pode ser acessada de qualquer *Web Browser* conectado à internet. A primeira interface pode ser vista na figura 4-1.

Figura 4-1: Site que mostra o consumo de energia dos aparelhos.



O segundo exemplo de interface com o usuário consiste de um aplicativo para dispositivos móveis. O aplicativo mostra o mesmo gráfico de consumo de energia, além de trazer botões que permitem ligar e desligar os equipamentos. Os botões têm as mesmas cores das curvas de consumo de energia dos objetos aos quais estão associados. Os botões trazem neles os nomes ***"Ligado"*** e ***"Desligado"***, indicando o estado do equipamento que eles controlam. A figura 4-2 mostra o segundo exemplo de interface gráfica.

Figura 4-2: Aplicativo para monitorar e controlar os equipamentos.



Nas interfaces, tanto os gráficos de consumo como os botões estão associados a URLs que disparam a execução de métodos no Web Service da central. A central por sua vez se comunica com os nodos por meio dos dispositivos Xbee.

4.4 TESTES

Os testes realizados são simples e visam demonstrar o funcionamento do sistema como um todo. Eles consistem do acionamento e desligamento dos equipamentos e análise da resposta. As figuras 4-3, 4-4 e 4-5 mostram alguns dos testes realizados.

O desligamento/acionamento dos equipamentos pode ser notado nas curvas de consumo. O sistema foi testado durante alguns dias em um ambiente residencial e demonstrou-se altamente estável e robusto, atendendo às expectativas.

Figura 4-3: Acionamento da lâmpada no segundo 78 e do ventilador no segundo 83.



Figura 4-4: Acionamento da lâmpada no segundo 35 e desligamento do ventilador no segundo 45.

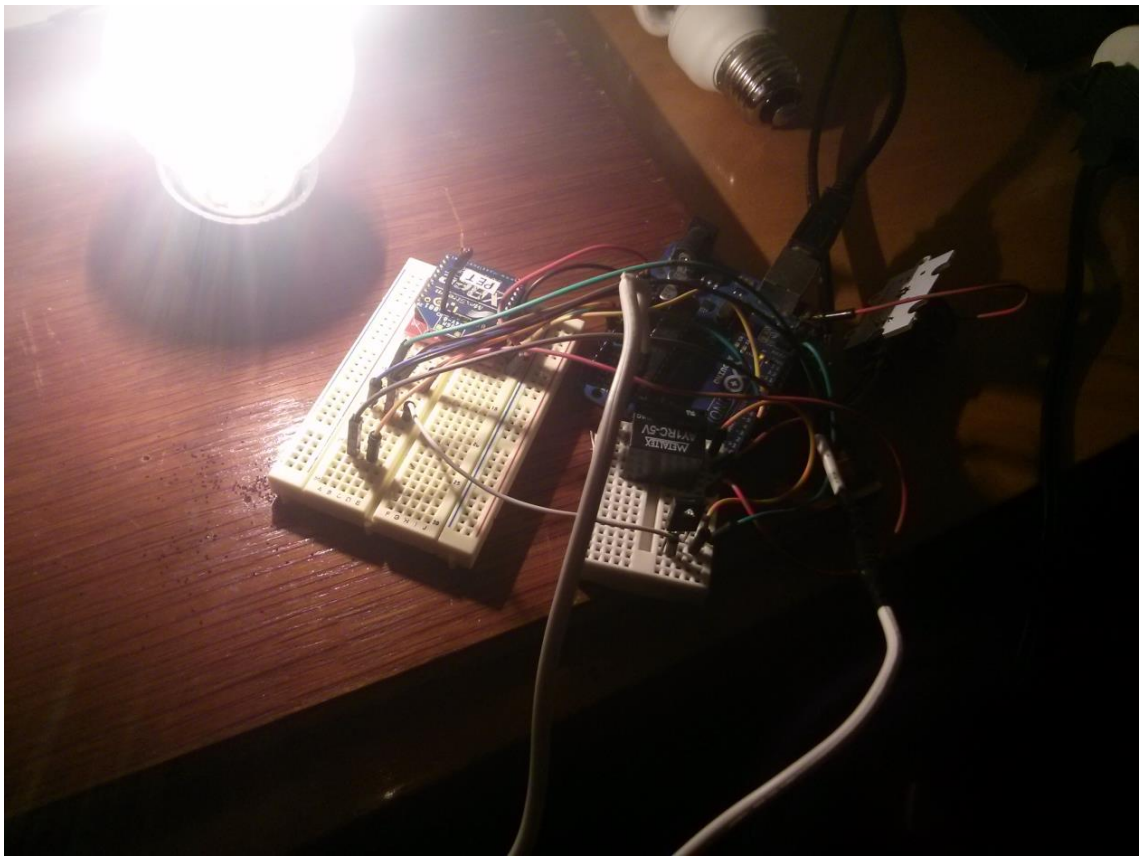


Figura 4-5: Ventilador (consumo 80W) mantido ligado. Lâmpada (consumo 30W) sendo desligada no segundo 87.



A figura 4-6 mostra a instalação do *hardware* de sensoriamento em uma lâmpada.

Figura 4-6: *Setup* do *hardware* de sensoriamento e acionamento em uma lâmpada.



5 CONCLUSÃO

Ao longo dos últimos anos a pesquisa em computação ubíqua teve avanços inegáveis. O tema deixou de existir apenas nas universidades e centros de pesquisa, para fazer parte do cotidiano da população. Dentro da computação ubíqua, as WSNs se destacam por terem a capacidade de trazer melhorias dos mais diversos tipos, além de constituírem uma possibilidade de solução para problemas fundamentais, por exemplo de cunho ambiental, biomédico, climático e econômico, que afetam diretamente a vida da espécie humana. (LOUREIRO et al., 2003)

No desenvolvimento deste Projeto de Graduação foi possível identificar premissas fundamentais para o projeto de WSNs como arquitetura de nodos sensores, protocolos de comunicação, tempo compartilhado em *software* embarcado, entre outros. Essas premissas, devidamente adaptadas ao domínio de aplicação, são relevantes ao projeto de qualquer WSN.

Foi implementada uma WSN centralizada, acessível via rede IP, para monitoramento do consumo de energia elétrica e controle do acionamento de equipamentos eletrônicos. A implementação pautou-se no estado da arte das tecnologias de sistemas embarcados para permitir a construção de um sistema computacional eficiente e de baixo custo, tendo por isso viabilidade de utilização prática.

A arquitetura implementada como prova de conceito pode ser doravante utilizada como referência no desenvolvimento de aplicações de baixo custo para sensoriamento, não só de consumo de energia, mas de qualquer tipo de grandeza física de interesse no domínio de aplicação em questão.

5.1 TRABALHOS FUTUROS

Neste Projeto de Graduação foram implementadas as funcionalidades básicas para permitir o monitoramento e o controle, local e remoto, de aparelhos eletrônicos. A

partir deste sistema básico abre-se um leque temas propícios para realização de pesquisa e implementação de melhorias, entre elas vale destacar:

- Estudo de um protocolo de autenticação e controle de acesso remoto, para garantir que o sistema não seja invadido por usuários maliciosos.
- Adição da funcionalidade de cômputo do consumo de energia para cargas de natureza altamente indutiva, considerando o efeito do defasamento entre a tensão e a corrente elétrica no valor da potência medida.
- Embora o Arduino funcione muito bem para os propósitos de uma WSN como a implementada, ele é um hardware de prototipação e sua principal vantagem é a facilidade de implementação. No processo de transformação do protótipo em produto, a substituição do Arduino por micro controladores e interfaces de hardware específicas permitiriam uma redução considerável no custo de produção de um módulo de sensoriamento e acionamento.
- Implementação de algoritmos de inteligência artificial para caracterização dos padrões de consumo energético nos pontos de interesse, permitindo identificação automática de possíveis anomalias no funcionamento de equipamentos.
- Criação de novos módulos que, dentro da mesma arquitetura de WSN, possam atuar, não somente no acionamento, mas também no controle do funcionamento de equipamentos, possibilitando uma vasta gama de novas aplicações (i.e. controle de temperatura de um ar condicionado, escolha do canal em um aparelho de TV etc).
- Aplicação do sistema a outros tipos de sensoriamento, além do de consumo de energia elétrica. O sensoriamento poderia ser usado, por exemplo, para controle automático de uma linha de produção, baseado em diversos tipos de dados físicos a serem adquiridos (i.e. temperatura, peso e velocidade).

- Estudo da viabilidade de agregação do 6LOWPAN, padrão que permitiria criar sobre a camada de enlace Zigbee a implementação do IPV6, permitindo que cada nodo da rede passe a possuir um IP.

6 REFERÊNCIAS BIBLIOGRÁFICAS

BERNDT, A. **Introdução a Redes de Sensores sem Fio (RSSF)**. Disponível em: <http://cct.uenp.edu.br/fernandolegore/Projetos/seguranca/Bibliografia/RFFS-introducao.pdf>. Acesso em: 2 jul. 2014.

DAWSON-HAGGERTY, S. et al. Energy Savings and the “Software-Defined” Building. **IEEE Design & Test of Computers**, v. 29, n. 4, p. 56–57, ago. 2012.

DIGI INTERNATIONAL. **XBee ® /XBee-PRO ® RF Modules**. Disponível em: <https://www.sparkfun.com/datasheets/Wireless/Zigbee/XBee-Datasheet.pdf>. Acesso em: 2 jul. 2014.

FALUDI, R. **Building Wireless Sensor Networks**. 1. ed. Sebastopol: O`REILLY, 2011. p. 1–321

LOUREIRO, A. A. F. et al. **Redes de Sensores Sem Fio**. Disponível em: <http://www.sensornet.dcc.ufmg.br/pdf/179_Loureiro_Nogueira_Ruiz_Mini_Nakamura_Figueiredo.pdf>. Acesso em: 2 jul. 2014.

LOUREIRO, A. A. F. **Redes de Sensores Sem Fio**. Disponível em: <http://www.ic.unicamp.br/~cmbm/desafios_SBC/loureiroredesensores.pdf>. Acesso em: 2 jul. 2014.

POTHUGANTI, K.; CHITNENI, A. A Comparative Study of Wireless Protocols : Bluetooth , UWB , ZigBee , and Wi-Fi. v. 4, n. 6, p. 655–662, 2014.

RUIZ, L. B. **MANÁ: Uma Arquitetura para Gerenciamento de Redes de Sensores Sem Fio**. [s.l.] Universidade Federal de Minas Gerais, 2003.

SILVA, F. A. et al. Tecnologia de Nós Sensores Sem Fio - Revista Controle e Instrumentação. p. 1–26, 2004.

THOMSON, S.; REKHTER, Y.; BOUND, J. **Dynamic Updates in the Domain Name System**. Disponível em: <http://tools.ietf.org/html/rfc2136>. Acesso em: 2 jul. 2014.

VAHID, F.; GIVARGIS, T.; MILLER, B. **Programming Embedded Systems: An Introduction to Time-Oriented Programming**. 4. ed. Lake Forert: UniWorld, 2012. p. 1–171

View publication stats